

Do It Yourself

인공지능 스피커와 Home IoT

v1.0

사용권한

본 문서에 대한 서명은 서명은 수행 및 유지 관리에 대한 책임이 있음을 인정하는 것임.

작성자 최의신 일자 2018-02-02

검토자 일자

본인은 서명으로써 업무활동 범위 내에서 사용될 것을 인가함.

승인자 일자

제.개정이력

버전	제.개정일자	제.개정 내용	제.개정자
v1.0	2018/02/02	제정	최의신

목 차

1. Prologue	4
2. 시스템 구성	5
3. 인공지능 스피커.....	7
3.1 하드웨어 구성	7
3.2 Android Things	8
3.3 gRPC Client.....	8
4. 서비스 서버	10
4.1 gRPC Server.....	10
4.2 Media Stream Serever.....	10
4.3 MySQL	13
4.4 MQTT Broker	14
5. IoT 디바이스.....	15
5.1 리모컨 데이터 전송	15
5.2 리모컨 코드 캡처	16
6. 설정파일.....	17
6.1 AiSpeakerEnv.java.....	17
6.2 db.properties	17
6.3 apps.conf	17
7. 소스파일.....	18
7. Epilogure	18

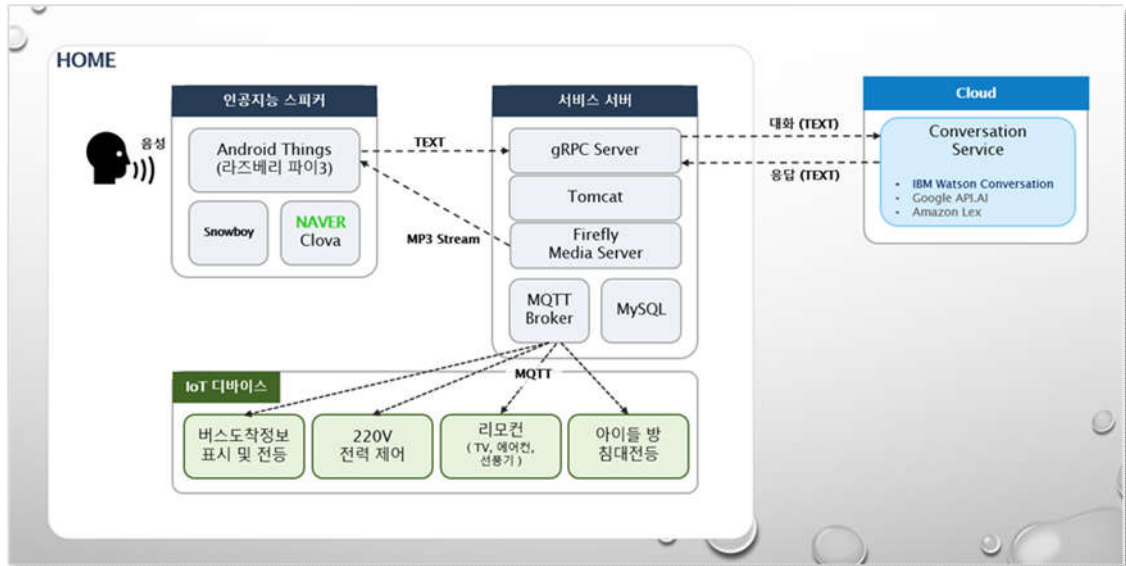
1. Prologue

인공지능의 발전이 빨라지면서 많은 제품들이 나오고 있다. Amazon echo를 필두로 Google, SK, NAVER, Kakao, KT 등에서 다양한 인공지능 스피커가 나오고 있다. 이것은 단순히 음성 인터페이스가 목표가 아니라 앞으로 열릴 Home IoT 시장에서 디바이스 허브 역할을 담당하도록 하는 것을 목표로 하는 것 같다. 실제로 LG의 경우 NAVER와 함께 Home IoT 상품을 출시한 상태이다.

그래서 이런 시대의 흐름에 맞게 이런 것을 만들어 보려고 한다. 이미 이전부터 시작했지만 한글을 처리하는 서비스가 없어서 이런 저런 테스트만 하던 차에 NAVER Clova와 Android Things를 알게 되었고, 이것을 이용하여 인공지능 스피커와 이전에 만든 IoT 디바이스를 결합하여 Home IoT를 실현해 보았다.

일단은 간단하게 제어할 수 있는 220V 전원제어와 리모컨을 이용한 제어만을 적용하였다. 여기서 재미있는 것은 리모컨을 이용하여 TV, 에어컨, 선풍기 등을 대화형으로 제어할 수 있다는 것이다.

2. 시스템 구성



- Android Things**
 안드로이드 씽즈는 사물인터넷(IoT) 기기 개발을 위한 플랫폼이다. 구글의 사물인터넷 플랫폼 브릴로(Brillo)를 업그레이드한 것으로 'Android Things'라는 이름을 새롭게 부여했다.
 현재 세 개의 디바이스에 사용 가능하며, 그 중에는 라즈베리파이 3가 있다.
- snowboy**
 현재 음성을 텍스트로 바꾸는 클라우드 서비스는 많이 존재한다. 하지만 클라우드를 사용하지 않고 구현하려면 그리 단순하지 않다. snowboy는 간단하게 학습을 하고, 이 결과를 이용하여 음성 인식을 수행한다. 그래서 이것을 이용하여 HOTWORD 감지를 수행하였다.
 현재는 'alexa'를 사용하지만 다른 단어를 학습하여 사용할 수 있다.
- NAVER Clova**
 한글 음성인식 및 음성합성 서비스를 제공한다. 하지만 제공되는 라이브러리가 아이폰, 안드로이드폰 만을 제공하여, 라즈베리파이 3에서는 사용할 수 없었다.
 하지만 안드로이드 씽즈를 이용하면 Clova 서비스를 사용할 수 있다.

- gRPC Server
인공지능 스피커에서 모든 서비스를 제공할 수 있다. 하지만 가급적 스피커의 기능을 단순화 하기 위해 서비스를 제공하는 서버를 따로 구성하였다. 그리고 여기에서 HTTP를 이용한 RESTful 서비스를 만들었다. 초기 접속 시 느린 현상을 빼면 그리 나쁘지 않았지만, 요즘 많은 관심을 받고 있는 gRPC를 이용하여 다시 작성하였다. 결과는 만족스러웠다.
- Tomcat
대화 서비스, DB 인터페이스, MP3 검색 등의 기능을 수행한다. 현재는 gRPC 서버에서 수행을 대신한다. 그리고 Firefly Media Server의 노래가 변경되면 여기에서 정보를 추출하여 검색에 사용할 테이블에 저장하는 기능도 담당한다.
- Firefly Media Server
얼마 전 잘 사용하던 음악 스트리밍 서버가 고장 나서 새로 구성한 서버이다. 재미있는 것은 MP3 정보를 SQLite3에 저장한다는 것이다. 이것을 이용하여 MP3 검색 기능을 구현하였다.
- MQTT Broker
이전에 만든 IoT 디바이스는 MQTT 프로토콜을 이용한다. 프로토콜 자체가 가볍고 또 사용할 수 있는 라이브러리도 많아 적합하다 생각된다. Apache Apollo를 사용하였다.
- MySQL
디바이스 정보, 리모컨 코드, MP3 정보 등을 저장한다.
- IoT 디바이스
디바이스 종류로는 버스 도착정보 표시, 침실 전등, 리모컨, 220V제어 등이 있으며, 특히 리모컨은 TV, Set-top Box, 에어컨, 선풍기 등을 제어할 수 있다. 즉 음성으로 명령을 내리면 그 결과로 리모컨을 이용해 디바이스를 제어한다.
- Conversation Service
어쩌면 인공지능 스피커에서 가장 중요한 부분이 아닐까 생각된다. 물론 모든 구성 요소가 중요하지만 문장을 해석하는 부분은 특히나 중요하다. 즉 사용자의 의도를 파악하고 무엇이 대상인지를 알아야 제어를 할 수 있기 때문이다. 이것을 위해서 이미 IBM, Google, Amazon 등의 기업이 클라우드 서비스를 제공한다. 여기에서는 IBM Watson Conversation을 사용한다.

3. 인공지능 스피커

3.1 하드웨어 구성



기능

- ① 동작 상태를 나타내는 LED
- ② 적외선 발광, 수광 LED
볼륨 UP/DOWN, 노래 STOP 기능을 수행
- ③ 마이크
- ④ RGB LED

구성


- A. 외관 - 비타민 통 활용
- B. 8개의 LED와 한 개의 RGB LED 제어 - 16CH PWM 모듈(PCA9685)
- C. AMP & 스피커 2개
- D. 라즈베리파이 3 와 USB 사운드 카드
- E. Reflective Object Sensor



A




B



C



D



E

Materials	Qty
라즈베리파이 3	2
USB-Serial (3.3v)	1
USB Sound Card	1
PCA9685	1
LED	8
RGB LED	1
QRD1114	2
마이크로폰	1
스피커	1
10K ohm	2
200 ohm	2

3.2 Android Things

2개의 라즈베리파이 중에서 한 개는 안드로이드 앱이 설치된다. 이것은 Android Studio를 이용하여 개발한다. 소스는 Git의 android 폴더에서 확인할 수 있다.

<https://developer.android.com/things/get-started/index.html>

3.3 gRPC Client

시스템 구성에서도 알 수 있듯이 인공지능 스피커에서 모든 것을 수행하지 않는다. 그러므로 일부 기능의 수행을 RESTful로 서버로 요청하여 처리하였다. 하지만 최초 수행 시 응답이 느려지는 문제가 있어 gRPC로 변경하였다. gRPC는 HTTP 2.0 기반으로 HTTP 1.0 보다 빠르며, 실제 적용해 보니 만족할 만한 성능을 보였다. gRPC로 구현한 서비스는 텍스트로 변환된 음성을 Watson Conversation으로 보내고, 그 결과를 받는다. Android Things에서는 Client 만 구현한다.

```
service AiService
{
    rpc say(AiServiceRequest) returns (AiServiceReply) ;
}

message AiServiceRequest {
    string payload = 1;
}

message AiServiceReply {
    string payload = 1;
}
```

1) AiServiceRequest

사용자 말한 내용을 문자로 전달한다. 즉 Speech Recognition 결과가 전달된다. 이것은 JSON 형식으로 다음과 같이 구성된다.

Field	설명	배열
request	요청객체	
- text	음성 텍스트	
- buttons	버튼입력 정보	Y
- pressId	버튼 ID	

2) AiServiceReply

Converation 서비스를 수행한 결과가 저장된다.

이것은 JSON 형식으로 다음과 같이 구성된다.

Field	설명	배열
response	응답객체	Y
- action	AI SPEAKER 동작	
- data	데이터 (OBJECT)	
returnCode		
errorString		

action	동작
speak	text를 음성으로 출력한다.
actuator	모터를 동작시킨다.
display	LCD or LED에 정보를 표시한다.
playMusic	음악을 재생한다.
playVideo	지정된 동영상을 재생한다.
irCommand	IR 리모콘 기능을 수행한다.

4. 서비스 서버

서비스 서버는 라즈베리파이를 이용하며, Watson Converation 서비스를 위한 gRPC Server, MP3 재생을 위한 Media Server, 노래 검색 및 IoT 디바이스 관리를 위한 MySQL Server 그리고 IoT 디바이스 통신을 위한 MQTT Broker 등이 설치된다.

개발은 eclipse를 이용하며, 소스는 Git의 server 폴더에서 확인할 수 있다.

4.1 gRPC Server

앞에서 설명한 AiService의 서버를 구현한다. 주 기능은 Watson Conversation의 응답에 따라 데이터베이스 조회, 웹 페이지 스크리핑, 외부 서비스 호출 등을 수행한 후 결과를 반환한다. 이것을 위해 간단한 framework를 이용한다.
보다 많은 정보는 다음을 참조한다.

```
https://developer.ibm.com/kr/developer-%EA%B8%B0%EC%88%A0-%ED%8F%AC%EB%9F%BC/2017/06/25/%EC%B1%97%EB%B4%87-%EA%B0%99%EC%9D%80-%EB%8C%80%ED%99%94%ED%98%95-%EC%84%9C%EB%B9%84%EC%8A%A4%EB%A5%BC-%EB%B9%A0%EB%A5%B4%EA%B2%8C-%EA%B0%9C%EB%B0%9C%ED%95%A0-%EC%88%98-%EC%9E%88%EB%8A%94-simple-frame/
```

4.2 Media Stream Serever

음악 스트리밍 서버는 mt-daapd를 이용하며, 구축 과정은 다음과 같다.

1) mt-daapd 구동에 필요한 패키지 설치

```
sudo apt-get install gawk gcc libsqlite0-dev libsqlite0 libgdbm-dev libid3tag0-dev  
avahi-daemon dpkg-dev libsqlite3-dev libgdbm-dev libflac-dev flac libvorbis-dev
```

2) mt-daapd 빌드 및 설치

소스 다운로드

```
wget http://pkgs.fedoraproject.org/repo/pkgs/mt-daapd/mt-daapd-svn-1696.tar.gz/42ba1f432bb88e18a8cb4ce0fc52eb64/mt-daapd-svn-1696.tar.gz
```

빌드 및 설치

```
tar xvzf mt-daapd-svn-1696.tar.gz

cd mt-daapd-svn-1696/

./configure --enable-oggvorbis --enable-flac --enable-sqlite3

make

sudo make install
```

3) 설정파일 mt-daapd.conf 수정

기본 설정파일 생성

```
sudo cp contrib/mt-daapd.conf /usr/local/etc/
```

/usr/local/etc/mt-daapd.conf 파일에서 다음의 항목을 수정

```
port = 3689

admin_pw = daapd

db_type = sqlite3

db_parms = /usr/local/var/cache/mt-daapd

mp3_dir = /home/orangepi/hdd/music

servername = MusicServer

password = daapd

extensions = .mp3, .m4a, .m4p, .flac, .ogg
```

항목	내용
db_type	음악파일의 태그정보를 sqlite3에 저장한다.
db_parms	지정된 디렉토리에는 song3db 파일이 존재하며, 모든 사용자 읽을 수 있도록 'r' 권한을 추가한다.
mp3_dir	음악 파일이 저장된 폴더를 지정한다.
extensions	flac, ogg 확장자를 추가한다.

4) 실행

```
/usr/local/sbin/mt-daapd
```

또는 /etc/rc.local 파일에 다음을 추가

```
su root -c '/usr/local/sbin/mt-daapd'&
```

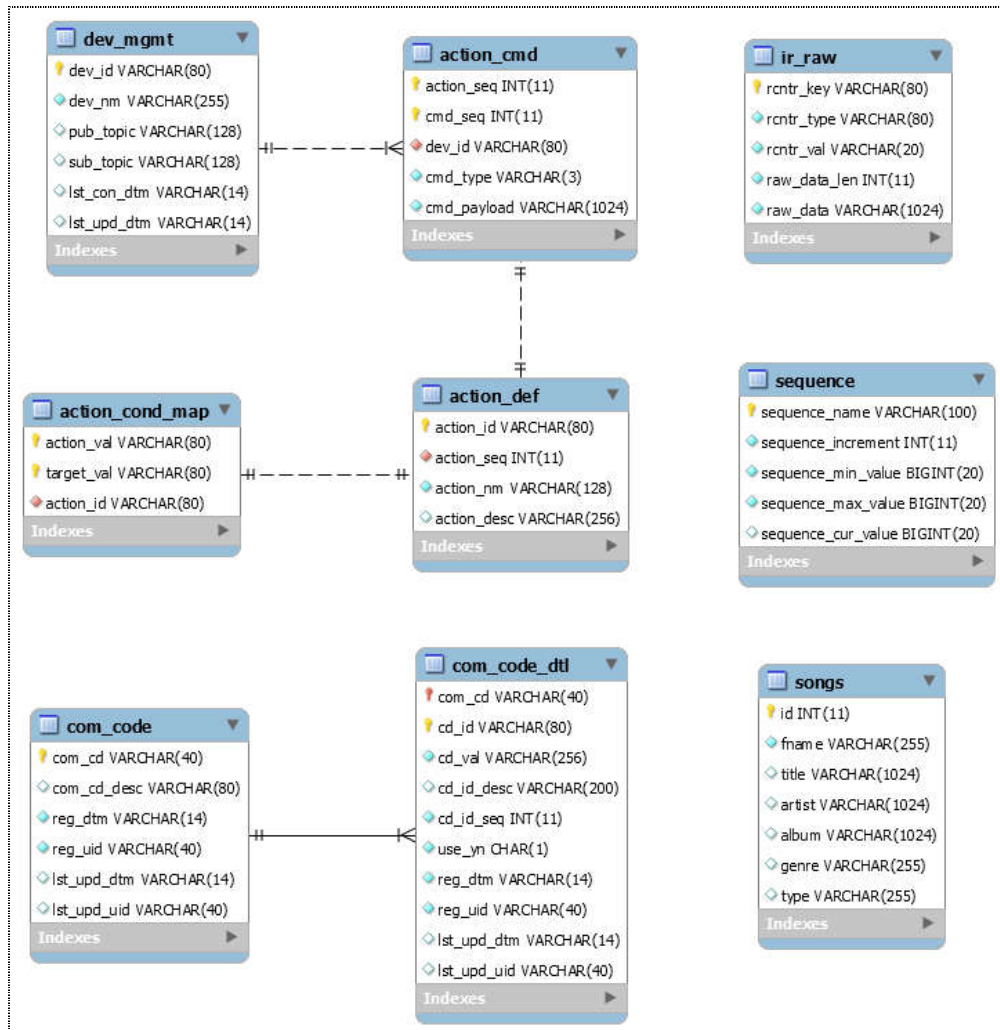
5) 동작확인

브라우저에서 [http://\[서버주소\]:3689](http://[서버주소]:3689)로 접속한다. 아이디는 admin이고, 비밀번호는 설정에 지정한 내용을 입력한다.

The screenshot shows the Firefly web interface. The top header includes the Firefly logo and the tagline "The best open-source media server for the Roku SoundBridge and iTunes". The left sidebar contains navigation links: "server status", "smart playlists", "configuration", "about firefly", and "thanks". The main content area is titled "Server Status". It features a table with columns "Service", "Status", and "Control". The services listed are "Bonjour" (Status: Stopped), "Firefly Media Server" (Status: Running), and "File scanner" (Status: Idle). The "Control" column contains buttons: "Stop Server" for Bonjour, "Start Scan" for Firefly Media Server, and "Start Full Scan" for File scanner. Below the table, there's a "Plugin" section with columns "Plugin" and "Version", listing "rsp", "ssc-script", and "daap" all at version "svn-1696". Further down, system statistics are shown: "Uptime" (41 days, 1 hour, 22 minutes, 10 seconds), "Songs" (3214), "Songs Served" (233), and "DB Version" (2). At the bottom, a "Client IP" section shows "192.168.0.62" and the action "Serving xml-rpc method".

4.3 MySQL

IoT 디바이스 정보, 리모콘 키 코드 데이터, 음악 데이터 등의 관리를 위해 MySQL을 사용한다.



테이블	설명
dev_mgmt	IoT 디바이스 정보를 관리한다.
action_cmd	디바이스가 수행할 명령의 형식(cmd_type) 및 데이터를 정의한다.
action_def	디바이스 별로 수행 가능한 명령을 정의한다.
action_cond_map	Conversation에서 해석된 결과와 디바이스 명령을 연결한다.
id_raw	리모컨의 IR 코드 값을 저장한다.
songs	노래(MP3) 태그 정보를 저장하며, 검색에 사용한다.
sequence	ID별로 유일한 순번을 생성한다.

스키마는 "myhomeai"로 지정하고, 사용자 및 비밀번호는 "homeai", "homeai"로 지정한다. 그리고 "myhomeai_dump.sql" 파일을 import 하여 설정을 완료한다.

MySQL 설치 및 설정은 다음을 참조한다.

<https://dev.mysql.com/doc/mysql-getting-started/en/>

4.4 MQTT Broker

IoT디바이스는 MQTT 프로토콜을 사용한다. 그러므로 MQTT Broker가 필요하다. 여기에서는 Apache Apollo 1.7.1을 사용하며, 다음과 같이 설정한다.

설정항목	설정값
MQTT 사용자 ID	homeai
MQTT 사용자 비밀번호	homeai
MQTT Broker Port	1883

상세한 정보는 다음을 참조한다.

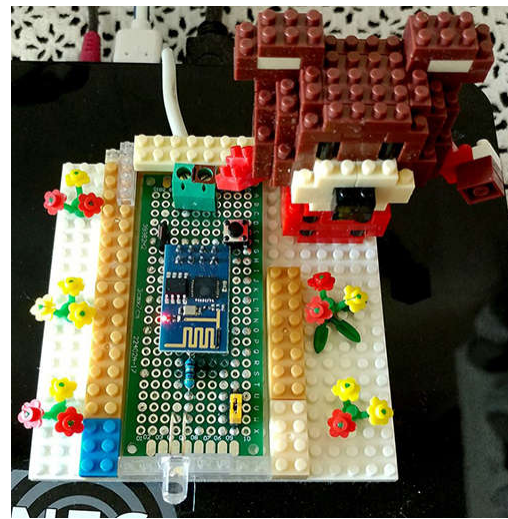
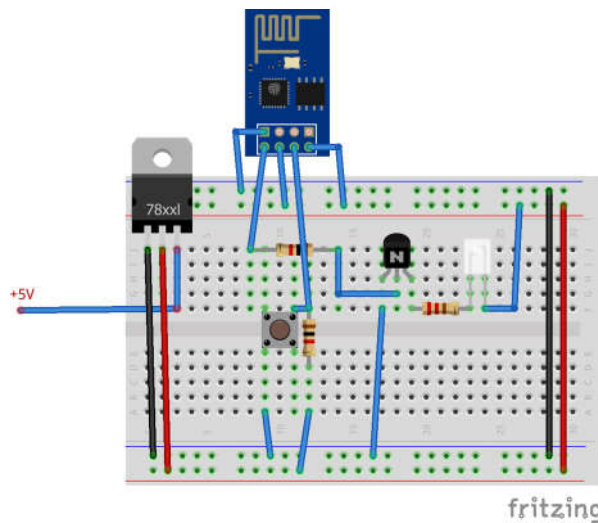
<https://activemq.apache.org/apollo/index.html>

5. IoT 디바이스

시스템 구성에는 많은 디바이스가 존재하지만 여기에서는 리모컨 데이터를 전송하는 디바이스만 설명한다.

5.1 리모컨 데이터 전송

1) 하드웨어



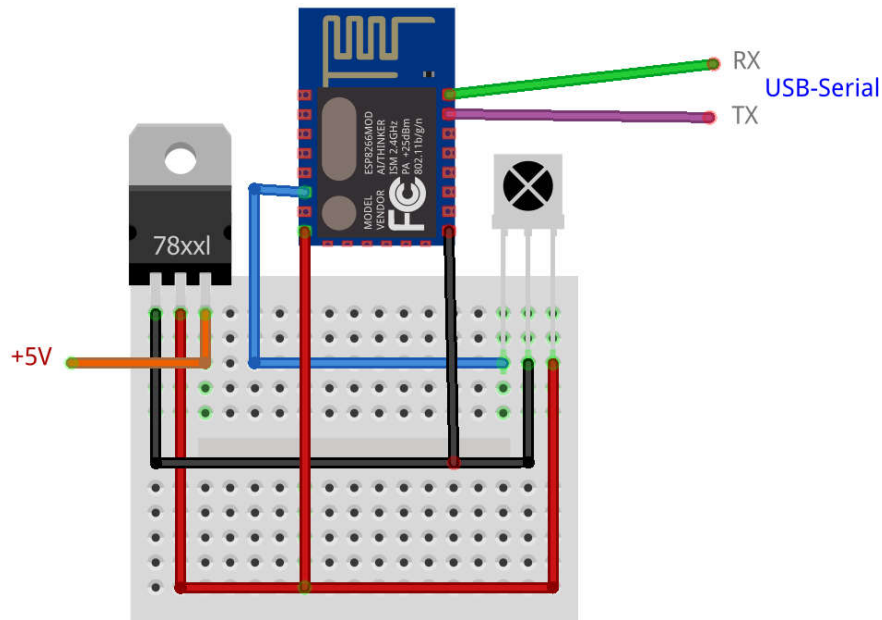
Materials	Qty
ESP8266-01	1
IR 발광 LED	1
PN2222A	1
200 ohm	1
1K ohm	1
PUSH Switch	1
AMS1117-3.3V	1

2) Arduino Code

iot 폴더의 IR_SEND.ino 파일 참조

5.2 리모콘 코드 캡처

1) 하드웨어



Materials	Qty
ESP8266-12	1
IR 수신 모듈	1
AMS1117-3.3V	1
USB-Serial (3.3V)	1

2) Arduino Code

iot 폴더의 IR_CAPTURE.ino 파일 참조

6. 설정파일

6.1 AiSpeakerEnv.java

폴더: android\MyHomeAiSpeaker\app\src\main\java\com\choi\hai

항목	설명
CLIENT_ID	Naver 어플리케이션 Client ID
CLIENT_SECRET	Naver 어플리케이션 Client Secret
RGB_LED	RGB LED가 연결된 PCA9685 IC의 채널 번호
PROGRESS_LED	8개의 LED가 연결된 PCA9685 IC의 채널 번호
GRPC_SERVER	서비스 서버의 IP 주소

6.2 db.properties

폴더: server\MyHome AI Project\resources

항목	설명
url	IP 주소를 서비스 서버의 IP로 변경
username	MySQL 사용자 ID
password	MySQL 사용자 비밀번호

6.3 apps.conf

폴더: server\MyHome AI Project\resources

항목	설명
mqtt.server	MQTT Broker 주소 (서비스 서버의 IP)
mqtt.id	사용자 ID
mqtt.pwd	사용자 비밀번호
songs.file	mt-daapd 서버의 songs3.db 파일의 경로
songs.server	mt-daapd 서버 IP
songs.username	admin
songs.password	mt-daapd 서버 설정에서 입력한 비밀번호
wcs.user	Conversation Service 사용자 ID
wcs.passwd	Conversation Service 비밀번호
wcs.workid	Conversation Workspace ID

7. 소스파일

소스 파일이 있는 Git URL은 다음과 같다.

```
https://github.com/TechOrgg/ai\_speaker\_iot.git
```

7. Epilogue

외형적인 부분, 마이크, 사운드 등 부족한 부분이 많다.

외형적인 부분은 제품으로 판매할 것이 아니므로 용서 가능하지만, 실 환경에서 사용하기에 마이크는 많이 부족하다. 그래서 Far field microphone과 같은 모듈이 필요하다. 그래야 3m 거리에서도 사용할 수 있다. 현재 아마존에 주문한 상태이며, 이것이 오면 교체할 예정이다. 그리고 사운드 또한 저렴하여 그리 만족스럽지 못하다. 그래서 사운드 전용 디바이스를 구성하고, 이것을 스피커와 연동하도록 할 계획이다.